**Experiment No 9: Pipelining Visualization**

|  |  |  |
| --- | --- | --- |
| **Sl No** | **Name** | **ID No** |
| **1** | **Dhruv Makwana** | **2019A3PS0381H** |

The aim of this experiment is to understand the basic principles of pipelining including the problems of data and branch hazards with the help of Ripes tool. Download the tool from CMS. Please pay attention to the demonstration given by the instructor.

Assume for all the following exercises the first instruction will get loaded in into IF stage 0th clock cycle (instead of 1st clock cycle as per our normal assumption)

**Exercise 9.1 Implement a simple RISC V program containing only add (add t0, t1, t2) instruction and explore (edit registers t1 and t2 to have non zero value.**

1. **For instruction add t0, t1, t2, briefly explain the things that happen (signal changes, Register changes and other effects if any) as the instruction goes through different pipeline stages.**

Answer: IF Stage: instruction code comes out of decoder

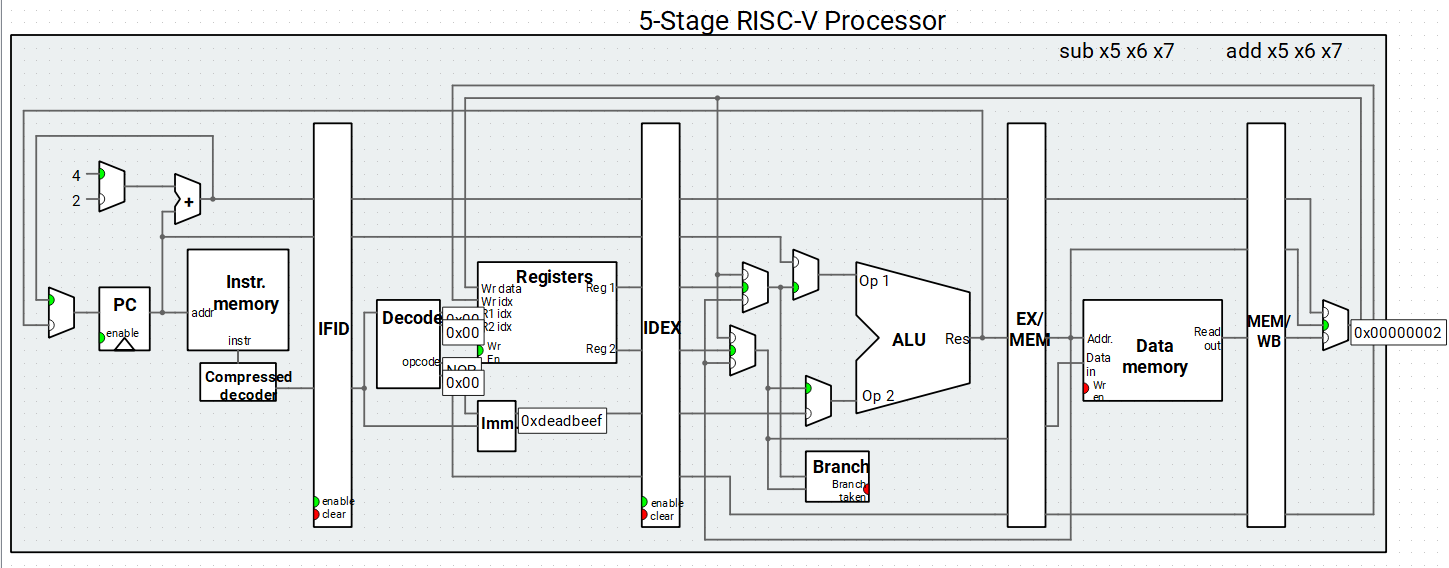
ID Stage: register file inputs and ALUop are decoded

EX Stage: ALU computes output

MEM Stage: wr\_en (memory) is 0, data is passed to next register

WB Stage: output of ALU is written to register file, wr\_en (reg\_file) is 1

1. **Copy the image of the pipeline at the end of 4th cycle (Show the value of important Signals).**

Answer: 

1. **How many clock cycles does it take for the result of the operation to be available in the destination register?**

Answer: 5

1. **In which pipeline stages do different arithmetic instructions differ?**

Answer: EX

1. **One stage is not used by arithmetic instructions. Which one? Why?**

Answer: **MEM**

**Exercise 9.2 Write the code below and test the Behaviour for 5-stage pipelined processor without forwarding**

**(Save non zero values in t1, t2, t3 and t4 register)**

**Code:**

**add t1, t2, t3**

**add t4, t1, t2**

**List the values stored in registers**

**t1: 1**

**t2: 2**

**t3: 3**

**t4: 4**

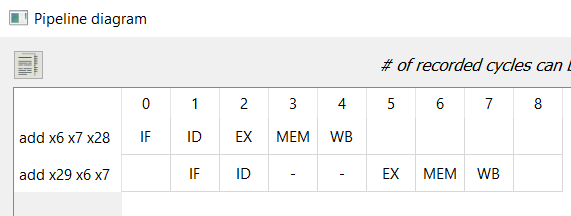
1. **At the end of which clock cycle will the destination register of the first add instruction (t1) gets updated?**

Answer: 5

1. **At the end of which clock cycle is the value of t1 needed in the second instruction?**

Answer: 3

1. **Copy the image of stage table after the completion of execution of second instruction.**

Answer: 

1. **What is the expected and actual value of t4 after the execution of second instruction?**

**Expected Value: 7**

**Actual Value: 3**

1. **What is the problem here? What is this kind of hazard called?**

Answer: **instruction 2 needs data from t1 before insutrction 1 has updated it. This is a data dependency hazard**

**Exercise 9.3 Write the code below and test the Behaviour for 5-stage pipelined processor with forwarding**

**(Save non zero values in t1, t2, t3 and t4 register)**

**Code:**

**add t1, t2, t3**

**add t4, t1, t2**

**List the values stored in registers**

**t1: 1**

**t2: 2**

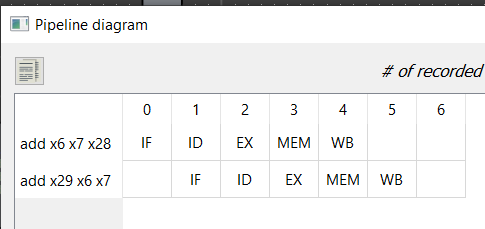
**t3: 3**

**t4: 4**

1. **At the end of which clock cycle will the destination register of the first add instruction, i.e. t1, gets updated?**

Answer: 5

1. **Copy the image of stage table after the completion of execution of second instruction.**

Answer: 

1. **What is the expected and actual value of t4 after the execution of second instruction?**

**Expected Value: 7**

**Actual Value: 7**

1. **How was the problem in the previous exercise resolved?**

Answer: via forwarding from MEM stage to EX stage

**Exercise 9.4 Write the code below and test the Behaviour for 5-stage pipelined processor without forwarding and without hazard detection**

**(Save non zero values in t1, t2 and t3. The default starting address of data memory is 0x10000000. In this case the data at address 0x10000000 will get loaded in t0, Check the memory to verify is 0x12340000 is stored.**

**Code:**

**.data**

**a: .word 0x12340000**

**.text**

**la t0, a**

**lw t1, 0(t0)**

**add t3, t1, t2**

**OR**

**(Save non zero values in t1, t2 and t3, let t0=0. In this case the data at address 0 will get loaded in t1, Check the memory to identify the data loaded 0th address [it will be equal to instruction code of first instruction i.e. lw t1, 0(t0)])**

**Code:**

**lw t1, 0(t0)**

**add t3, t1, t2**

**List the values stored in registers**

**t1: 1**

**t2: 2**

**t3: 3**

**List the data value stored in memory location pointed by t0: 0x0002a303**

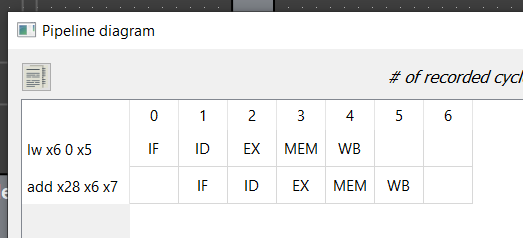
1. **At the end of which clock cycle will the destination register of the lw, i.e. t1, gets updated?**

Answer: 5

1. **At the end of which clock cycle is the value of t1 needed in the add t3, t1, t2 instruction?**

Answer: **3**

1. **Copy the image of stage table after the completion of execution of add t3, t1, t2 instruction.**

Answer: 

1. **What is the expected and actual value of t3 after the execution of add t3, t1, t2 instruction?**

**Expected Value: 0x0002a305**

**Actual Value: 3**

1. **What is the problem here? What is this kind of hazard called?**

Answer: The data needed by instruction 2 is on cycle 3 whereas the data is coming on cycle 5. This is a data hazard.

**Exercise 9.5 Write the code below and test the Behaviour for 5-stage pipelined processor with forwarding and without hazard detection**

**(Save non zero values in t1, t2 and t3. The default starting address of data memory is 0x10000000. In this case the data at address 0x10000000 will get loaded in t0, Check the memory to verify is 0x12340000 is stored.**

**Code:**

**.data**

**a: .word 0x12340000**

**.text**

**la t0, a**

**lw t1, 0(t0)**

**add t3, t1, t2**

**OR**

**(Save non zero values in t1, t2 and t3, let t0=0. In this case the data at address 0 will get loaded in t1, Check the memory to identify the data loaded 0th address [it will be equal to instruction code of first instruction i.e. lw t1, 0(t0)])**

**Code:**

**lw t1, 0(t0)**

**add t3, t1, t2**

**List the values stored in registers**

**t1: 1**

**t2: 2**

**t3: 3**

**List the data value stored in memory location pointed by t0: 0x0002a303**

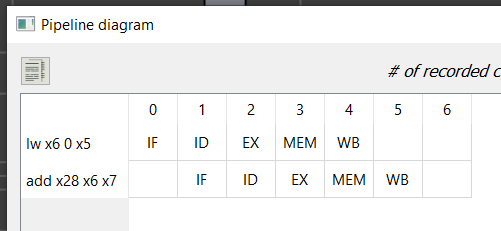
1. **At the end of which clock cycle will the data, that is to be written in to t1, be ready?**

Answer: 3

1. **At the end of which clock cycle is the value of t1 needed in the add t3, t1, t2 instruction?**

Answer: **3**

1. **Copy the image of stage table after the completion of execution of add t3, t1, t2 instruction.**

Answer: 

1. **What is the expected and actual value of t3 after the execution of add t3, t1, t2 instruction?**

**Expected Value: 0x0002a305**

**Actual Value: 2**

1. **Why was the problem not resolved even after adding forwarding Unit?**

Answer: Because we need an additional stall for the data to be ready in the MEM stage.

**Exercise 9.5 Write the code below and test the Behaviour for 5-stage pipelined processor with forwarding and with hazard detection**

**(Save non zero values in t1, t2 and t3. The default starting address of data memory is 0x10000000. In this case the data at address 0x10000000 will get loaded in t0, Check the memory to verify is 0x12340000 is stored.**

**Code:**

**.data**

**a: .word 0x12340000**

**.text**

**la t0, a**

**lw t1, 0(t0)**

**add t3, t1, t2**

**OR**

**(Save non zero values in t1, t2 and t3, let t0=0. In this case the data at address 0 will get loaded in t1, Check the memory to identify the data loaded 0th address [it will be equal to instruction code of first instruction i.e. lw t1, 0(t0)])**

**Code:**

**lw t1, 0(t0)**

**add t3, t1, t2**

**List the values stored in registers**

**t1: 10**

**t2: 20**

**t3: 30**

**List the data value stored in memory location pointed by t0: 0x0002a303**

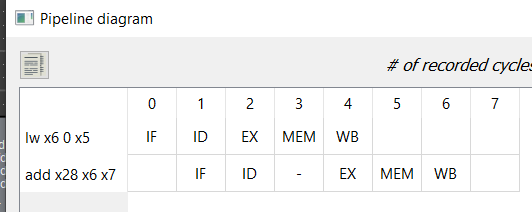
1. **At the end of which clock cycle will the data, that is to be written in to t1, ready?**

Answer: 3

1. **At the end of which clock cycle is the value of t1 needed in the add t3, t1, t2 instruction (including stall)?**

Answer: 4

1. **Copy the image of stage table after the completion of execution of add t3, t1, t2 instruction.**

Answer: 

1. **What is the expected and actual value of t3 after the execution of add t3, t1, t2 instruction?**

**Expected Value: 0x0002a323**

**Actual Value: 0x0002a323**

1. **How was the problem (encountered in previous two exercises) resolved?**

Answer: By delaying the second instruction’s EX stage by one cycle.

**Exercise 9.6 Write the code below and test the Behaviour for 5-stage pipelined processor with forwarding and with hazard detection (this data path uses static Branch NOT TAKEN scheme)**

**Code**

**beq t1, t0, L1**

**sub t4, t3, t2**

**add t1, t2, t3**

**add t1, t2, t3**

**add t1, t2, t3**

**L1: sub t4, t2, t3**

**Case 1 (t0==t1) (Store Non zero values in t0, t1, t2, t3, t4 and make sure t0 is equal to t1)**

**List the values stored in registers**

**t0: 5**

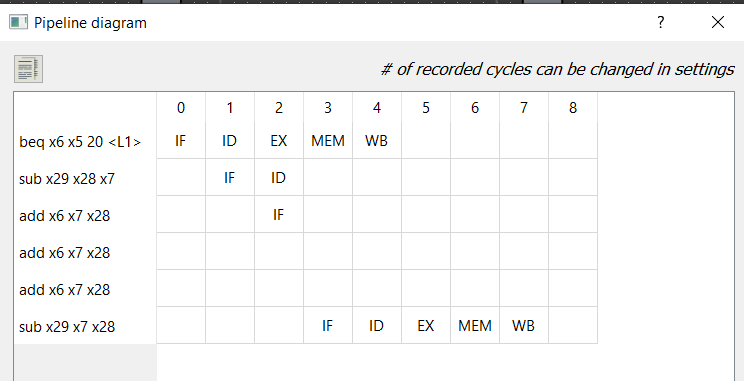
**t1: 5**

**t2: 10**

**t3: 10**

**t4: 4**

1. **Copy the image of stage table after the completion of execution.**

Answer: 

1. **Explain the pipelined processor operation for this case.**

Answer: The branch is predicted as not taken, so the next 2 instructions in memory proceed to their IF and ID stages. However, after the branch reaches its EX stage, we find that the condition was true and the branch had to be taken. So, the instructions that were previously loaded are flushed, and L1 goes into its IF stage.

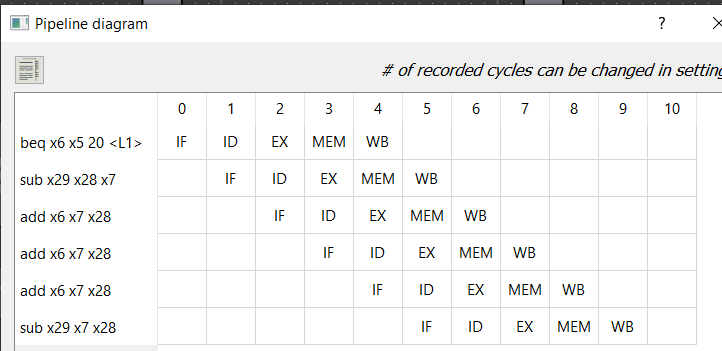
1. **What is the expected and actual value of t4 after the completion of execution?**

**Expected Value: 0**

**Actual Value: 0**

**Case 1 (t0! = t1) (Store Non zero values in t0, t1, t2, t3, t4 and make sure t0 is not equal to t1)**

1. **Copy the image of stage table after the completion of execution.**

Answer: 

1. **Explain the pipelined processor operation for this case.**

Answer: Operation proceeds sequentially, in the same order that the intructions are loaded into the text segment. (this is because the branch was predicted successfully)

1. **What is the expected and actual value of t4 after the completion of execution?**

**Expected Value: 0**

**Actual Value: 0**

**General**

1. **List the concepts you learnt from this experiment (Conclusions/Observations)**

Answer: We learned how to analyze the riscV pipeline and see the effects of hazard detection and forwarding.